C++ ASSIGNMENT

1.Ques :Write a program to apply binary search in array sorted in decreasing order.

Ans:
```
#include<iostream>
using namespace std;
int main(){
    int v[]={5,4,3,2,1};
    int n=5;
    int x;
    cout<<"Enter the target:";
    cin>>x;
    int lo=n-1;
    int hi=0;
    bool flag=false;
    while(hi<=lo){
        int mid=hi+(lo-hi)/2;
        if(v[mid]==x){
            cout<<mid;
            flag=true;
            break;
        }
        else if(v[mid]<x) lo=mid+1;
        else hi=mid-1;
    }
    if(flag==false) cout<<-1;
}
```

2.Ques :You have a sorted array of infinite numbers, how would you search an element in the array?

Ans: using binary search.

3.Ques :You are given an m x n integer matrix matrix with the following two properties:

Each row is sorted in non-decreasing order.

The first integer of each row is greater than the last integer of the previous row.

Given an integer target , return true if target is in matrix or false otherwise.

You must write a solution in O(log(m * n)) time complexity.

Example 1:

Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3

Output: true

Example 2:

Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 13

Output: false

Ans:
```
#include<iostream>
#include<vector>
using namespace std;
int main(){
    int t;
    cin>>t;
    for(int i=0;i<t;i++){
        vector<vector<int>> v
 {
   {1, 3, 5,7},
   {10,11,16,20},
   {23,30,34,60}
 };
 int n=v.size();
```

```
    int m=v[0].size();
    int x;
    cout<<"Enter the target:";
    cin>>x;
    bool flag=false;
    for(int i=0;i<n;i++){
        int lo=0;
        int hi=m-1;
      while(lo<=hi){
        int mid=lo+(hi-lo)/2;
        if(v[i][mid]==x){
            flag=true;
            break;
        }
        else if(v[i][mid]<x) lo=mid+1;
        else hi=mid-1;
    }
    }
    if(flag==false) cout<<"false";
    else cout<<"true";
    }

}
```

4.Ques:There is an integer array nums sorted in non-decreasing order (not necessarily with distinct values). Before being passed to your function, nums is rotated at an unknown pivot index k (0 <= k< nums.length) such that the resulting array is [nums[k], nums[k+1], ..., nums[n-1],nums[0], nums[1], ..., nums[k-1]] (0-indexed). For example,[0,1,2,4,4,4,5,6,6,7] might be rotated at pivot index 5 and

become [4,5,6,6,7,0,1,2,4,4] .
Given the array nums after the rotation and an integer target , return true if target is in
nums , or false if it is not in nums .
You must decrease the overall operation steps as much as possible.
Example 1:
Input: nums = [2,5,6,0,0,1,2], target = 0
Output: true
Example 2:
Input: nums = [2,5,6,0,0,1,2], target = 3
Output: false

Ans:
```
#include<iostream>
#include<vector>
using namespace std;
int main(){
    int n;
    int arr[n]={1,2,4,0,1,2,5,9};
    n=8;
    int x;
    cout<<"Enter the target:";
    cin>>x;
    bool flag=false;
    int lo=0;
    int hi=n-1;
    while(lo<=hi){
        int mid=lo+(hi-lo)/2;
        if(arr[mid]==x){
            flag=true;
            break;
        }
        else if((arr[lo]==arr[mid]) && (arr[hi]==arr[mid])){
            lo++;
            hi--;
        }
        else if(arr[lo]>arr[mid]){
            if((arr[mid]<x)&&(arr[hi]>=x)) lo=mid+1;
            else hi=mid-1;
        }
        else{
            if((arr[mid]>x)&&(arr[lo]<=x)) hi=mid-1;
            else lo=mid+1;
        }
    }
    if(flag==true) cout<<"true";
    else  cout<<"false";
}
```